The Journal of Open Source Software

Elasticipy: A Python package for linear elasticity and tensor analysis

Dorian Depriester [1¶] **and Régis Kubler** [1]

1 Arts et Métiers Institute of Technology, MSMP, Aix-en-Provence, F-13617, France ¶ Corresponding author

DOI: 10.21105/joss.07940

Software

- Review
- Repository
- Archive

Editor: Sarath Menon

Reviewers:

- @hakonanes
- @AHartmaier

Submitted: 23 January 2025
Published: 26 November 2025

Summary

Elasticipy is a Python library designed to streamline computation and manipulation of elasticity tensors for materials and crystalline materials, taking their specific symmetries into account. It provides tools to manipulate, visualise, and analyse tensors –such as stress, strain and stiffness tensors– simplifying workflows for materials scientists and engineers.

Statement of Need

In continuum mechanics, the deformation of a material is described by the second-order strain tensor (usually denoted ε) whereas the stress is described by the second-order Cauchy's stress tensor ($\boldsymbol{\sigma}$). Under the linear elasticity assumption, the relationship between the elastic strain ε and $\boldsymbol{\sigma}$, known as the generalised Hooke's law, is given through the fourth-order stiffness tensor $\boldsymbol{C}$ with:

$$\boldsymbol{\sigma} = \boldsymbol{C} : \boldsymbol{\varepsilon}$$

where ":" denotes the tensor product contrated twice, so that:

$$\sigma_{ij} = C_{ijk\ell}\varepsilon_{k\ell}$$

In order to simplify the above equations, one usually uses the so-called Voigt notation, which reads:

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{13}\\ \sigma_{12}\end{bmatrix} = \begin{bmatrix} C_{1111} & C_{1122} & C_{1133} & C_{1123} & C_{1113} & C_{1112}\\ & C_{2222} & C_{2233} & C_{2223} & C_{2213} & C_{2212}\\ & & C_{3333} & C_{3323} & C_{3313} & C_{3312}\\ & & & C_{2323} & C_{2313} & C_{2312}\\ & \text{sym.} & & & C_{1313} & C_{1312}\\ & & & & & C_{1212}\end{bmatrix} \begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ 2\varepsilon_{23}\\ 2\varepsilon_{13}\\ 2\varepsilon_{12}\end{bmatrix}$$

The values of $\boldsymbol{C}$ depend on the material, whereas its pattern (set of zero-components, or linear relationships between them) depends on the material's symmetry (Nye, 1985), as outlined in Figure 1.

Pymatgen (Ong et al., 2013) provides some built-in functions to work on strain, stress and elasticity but lacks some functionalities about the tensor analysis. Conversely, Elate (Gaillac et al., 2016) is a project dedicated to analysis of stiffness and compliance tensors (e.g. plotting directional engineering constants, such as Young modulus). It is implemented in the Materials Project (Jain et al., 2013). AnisoVis (Healy et al., 2020) is similar to Elate, but works on MATLAB®.

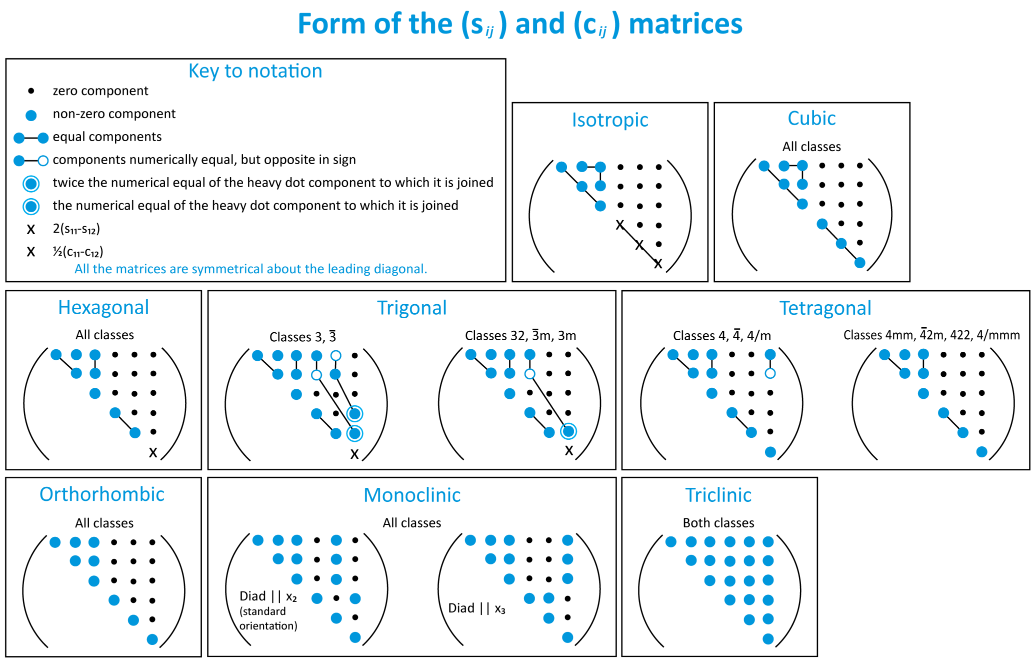
Form of the (s_ij) and (c_ij) matrices
Key to notation
zero component
non-zero component
equal components
components numerically equal, but opposite in sign
twice the numerical equal of the heavy dot component to which it is joined
the numerical equal of the heavy dot component to which it is joined
x 2(s11-s12)
x ½(c11-c12)
All the matrices are symmetrical about the leading diagonal.
Isotropic
Cubic
All classes
Hexagonal
All classes
Trigonal
Classes 3, 3̄
Classes 32, 3̄m, 3m
Tetragonal
Classes 4, 4̄, 4/m
Classes 4mm, 4̄2m, 422, 4/mmm
Orthorhombic
All classes
Monoclinic
All classes
Diad || x2 (standard orientation)
Diad || x3
Triclinic
Both classes

Figure 1: Patterns of stiffness and compliance tensors of crystals, depending on their symmetry (Nye, 1985). With courtesy of Pr. Pamela Burnley.

Therefore, the purpose of Elasticipy is to combine the functionalities of Pymatgen and Elate into a consistent Python project dedicated to continuum mechanics. Its aim is to propose an easy-to-use and efficient tool with the following features:

- intuitive Python-based APIs for defining and manipulating second- and fourth-order tensors, such as strain, stress and stiffness;
- support for standard crystal symmetry groups (Nye, 1985) to facilitate the definition of stiffness/compliance components;
- visualisation tools for understanding directional elastic behaviour (Young modulus, shear modulus and Poisson ratio);
- a collection of built-in methods to easily and efficiently perform fundamental operations on tensors (rotations (Depriester, 2019), products, invariants, statistical analysis etc.);
- averaging techniques, such as Voigt–Reuss–Hill (Hill, 1952), for textured and non-textured polycrystalline multiphased aggregates.

In order to evidence some of these features, Figure 2.a) illustrates the directional Young modulus of copper (Cu) single crystal as a 3D surface, whereas Figure 2.b) shows the same values as a pole figure (Lambert projection). In Figure 2.c), the Young modulus of a polycrystalline Cu displaying a perfect $[001]$ fibre texture has been estimated with different averaging methods (namely Voigt, Reuss and Hill (Hill, 1952)), then plotted as orthogonal sections.

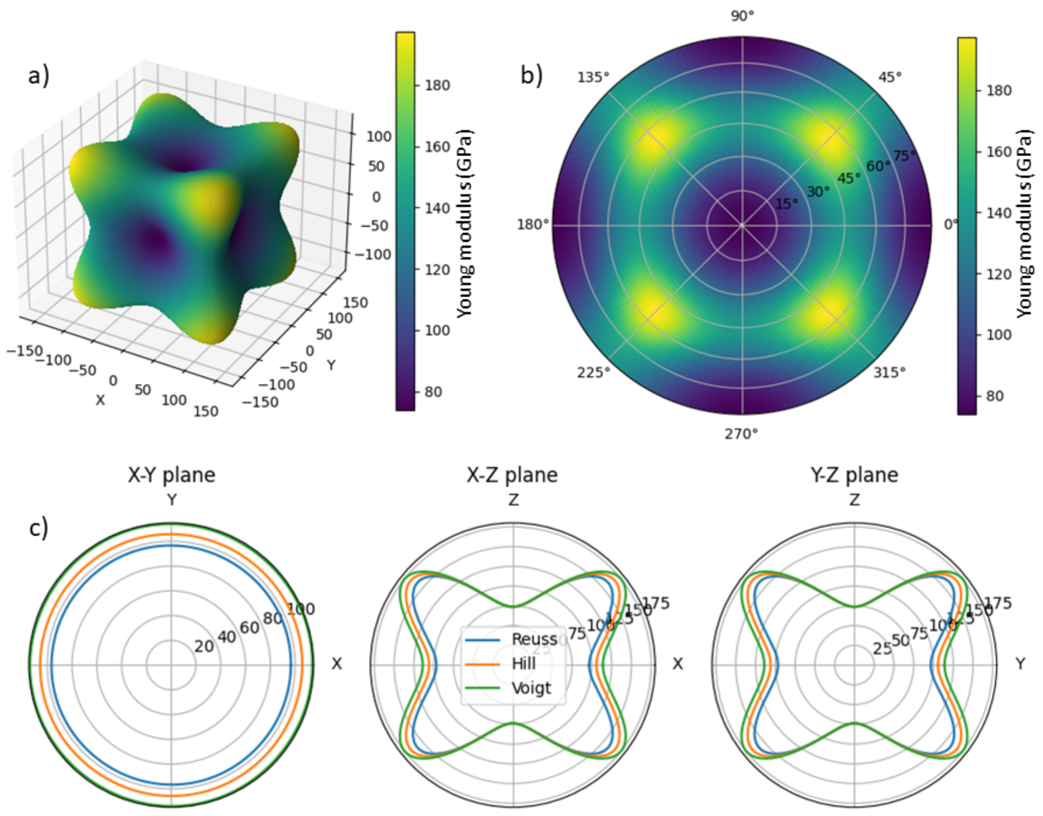

Figure 2: Young modulus (GPa) of Cu single crystal as a 3D surface (a) or a pole figure (b); Young modulus of Cu polycrystal with $[001]$ fibre texture, plotted in three orthogonal sections, depending on the averaging method (c).

Elasticipy also introduces the concept of *tensor arrays*, in a similar way as in MTEX (Mainprice et al., 2011), allowing to process several tensors at once with simple and highly efficient commands. In order to highlight the performances of Elasticipy, Figure 3 shows the wall-time required to perform two basic operations on tensors (namely, apply the generalised Hooke's law and compute the von Mises equivalent stress) as functions of the number of considered tensors. This demonstrates that, when processing large datasets of tensors ($n > 10^3$), basic tensor operations are 1 to 2 orders of magnitude faster in Elasticipy compared to Pymatgen. These performance gains are achieved by leveraging NumPy's array broadcasting capabilities (Harris et al., 2020). However, as tensor algebra is not the primary focus of Pymatgen, Elasticipy is designed to complement rather than replace it. Elasticipy supports seamless conversion between its own data structures and those of Pymatgen, allowing users to integrate both tools and benefit from Pymatgen's extensive features beyond tensor analysis. Elasticipy is also compatible with Orix (Johnstone et al., 2020), a Python library for analysing orientations and crystal symmetry.

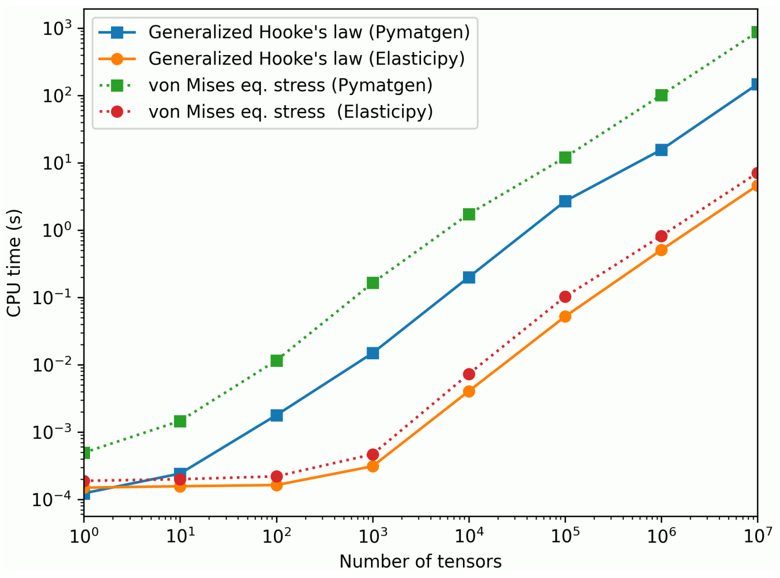

Figure 3: Performance comparison between Elasticipy and pymatgen.

Possible extensions

It is worth mentioning that Elasticipy provides a full framework for working on tensors, allowing to extend the analyses to other averaging methods (e.g. self-consistent models), possibly beyond linear elasticity problems (e.g. plasticity) with ease. It already implements thermal expansion.

Usage

This section presents the syntaxes of few basic operations performed with Elasticipy v4.2.0.

Plot directional engineering constants

Figure 2.a) and b) were rendered with the following syntax:

```
from Elasticipy.tensors.elasticity import StiffnessTensor
C = StiffnessTensor.cubic(C11=186, C12=134, C44=77)
E = C.Young_modulus
fig, _ = E.plot3D(n_phi=500, n_theta=500)
fig.show()
fig, _ = E.plot_as_pole_figure()
fig.show()
```

Create an array of rotated stiffness tensors and compute average

When considering a finite set of orientations, an array of stiffness tensors can be built to account for the rotations:

```
from scipy.spatial.transform import Rotation
import numpy as np
n = 10000
phi1 = np.random.random(n)*2*np.pi  # Random sampling from 0 to 2pi
Euler_angles = np.array([phi1,  np.zeros(n),  np.zeros(n)]).T # Fibre texture
```

```python
rotations = Rotation.from_euler('ZXZ', Euler_angles) # Bunge-Euler angles
C_rotated = C * rotations # n-length tensor array
```

Then, the Voigt–Reuss–Hill average (Hill, 1952) can be computed from the tensor array:

```python
C_VRH = C_rotated.Hill_average()
```

Finally, the corresponding Young moduli can be plotted in orthogonal sections, as shown in Figure 2.c), with:

```python
fig, ax = C_VRH.Young_modulus.plot_xyz_sections()
fig.show()
```

Arrays of stress/strain tensor

Efforts have been made to provide out-of-the-box simple syntaxes for common operations. For example, the following will create a tensor array corresponding to evenly-spaced strain along $[1, 0, 0]$ axis:

```python
from Elasticipy.tensors.stress_strain import StrainTensor
m = 1000  # length of tensor array
mag = np.linspace(0, 0.1, m)  # Strain magnitude
strain = StrainTensor.tensile([1, 0, 0], mag)
```

Given the stiffness tensor `C` (see above), one can compute the corresponding stress array with:

```python
stress = C * strain
```

Finally, `stress.vonMises()` returns an array of length `m` and data type `float64`, providing all the corresponding von Mises equivalent stresses.

References

Depriester, D. (2019). *Mean elastic properties of isotropic polycrystals: Comparison between averaging approximations with numerical computations*. ResearchGate. https://doi.org/10.13140/RG.2.2.36348.59521/3

Gaillac, R., Pullumbi, P., & Coudert, F.-X. (2016). ELATE: An open-source online application for analysis and visualization of elastic tensors. *Journal of Physics: Condensed Matter*, *28*(27), 275201. https://doi.org/10.1088/0953-8984/28/27/275201

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Healy, D., Timms, N. E., & Pearce, M. A. (2020). The variation and visualisation of elastic anisotropy in rock-forming minerals. *Solid Earth*, *11*(2), 259–286. https://doi.org/10.5194/se-11-259-2020

Hill, R. (1952). The elastic behaviour of a crystalline aggregate. *Proceedings of the Physical Society. Section A*, *65*(5), 349. https://doi.org/10.1088/0370-1298/65/5/307

Jain, A., Ong, S. P., Hautier, G., Chen, W., Richards, W. D., Dacek, S., Cholia, S., Gunter, D., Skinner, D., Ceder, G., & Persson, K. A. (2013). Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, *1*(1), 011002. https://doi.org/10.1063/1.4812323

Johnstone, D. N., Martineau, B. H., Crout, P., Midgley, P. A., & Eggeman, A. S. (2020). Density-based clustering of crystal (mis)orientations and the orix Python library. *Journal of*

Applied Crystallography, *53*(5), 1293–1298. https://doi.org/10.1107/S1600576720011103

Mainprice, D., Hielscher, R., & Schaeben, H. (2011). Calculating anisotropic physical properties from texture data using the MTEX open-source package. *Geological Society, London, Special Publications*, *360*(1), 175–192. https://doi.org/10.1144/SP360.10

Nye, J. F. (1985). *Physical properties of crystals: Their representation by tensors and matrices*. Oxford university press.

Ong, S. P., Richards, W. D., Jain, A., Hautier, G., Kocher, M., Cholia, S., Gunter, D., Chevrier, V. L., Persson, K. A., & Ceder, G. (2013). Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, *68*, 314–319. https://doi.org/10.1016/j.commatsci.2012.10.028